Luke Demi

Critical Thinking Questions

1. Suppose your application/dll were deployed in an enterprise environment. How could an attacker leverage this method of linking as a mechanism of persistence? Describe what the complete attack would look like, from start to finish.
    a. The attacker would first have to gain access to the enterprises's network, through a variety of different methods such as phishing email scams or physical system attacks. Then, the attacker would have to replace my original DLL with their own malicious DLL, without the enterprise knowing. When the DLL is eventually used by whatever application it is being used for, the attackers malicious code will be loaded, therefore executing the code.
2. What steps could an attacker take to minimize their chances of being detected due to the application functionality being disrupted?
    a. An attacker could do a couple of things to try and avoid detection of the malicious code, including functionality mimicing of the original DLL and the intentional deployment of the malicious DLL to specific systems in the network. By making the malicious DLL as close as possible to the functionality of original DLL, it will be nearly impossible for network administrators to detect. Also, by choosing specific systems on the network to target first with the malicious DLL, the attacker can slowly rollout the impact of the malware over time, which would minimize the chance of detection and make it harder for administrators to prevent.
3. What artifacts would be impossible for an attacker to change?
    a. Two things that are impossible for an attacker to change would be file timestamps and code signatures. When the original DLL is swapped out there will be a change in the file timestamp which is impossible to forge. Also, some DLL's use certificates in order to preserve integrity and provide authenticity for the code, and this signature will also be impossible by an attacker to change.
4. Describe how your application could be rewritten to prevent these attacks. Be as specific as possible, although you do not need to provide actual code.
    a. The first thing I could do to prevent DLL attacks would be to simply make it a static library rather than a dynamic library, but since this change would increase the compilation time for applications it would not be ideal. Another thing I could possibly do is create a separate program to save a hash of the original DLL file and compare that against the hash of the current DLL file, which would show if it has been tampered with. This would notify the administrators of the network whenever the DLL has been tampered with and serve as an integrity check for the DLL. Finally, I could digitally sign my DLL with a code signing certificate in order to prove that the DLL is authentic and has not been altered with.

5. What recommendations does Microsoft provide for mitigating DLL-related vulnerabilities? Be sure to cite your sources.
    a. Microsoft recommends a variety of different techniques in order to mitigate DLL-related vulnerabilities. First, Microsoft recommends the usage of process monitor, which can deeper examine the DLL load operations in an application. This will not be able to prevent DLL injection from happening, but will make it easier to detect whenever it does happen so it can be fixed. Microsoft also recommends applying the principle of least privilege, so if a DLL vulnerability does pop up then the attacker can't do much, since the attacker will not have many permissions. A final recommendation is to simply update all of your software as often as possible, since patches remove some of these vulnerabilities.

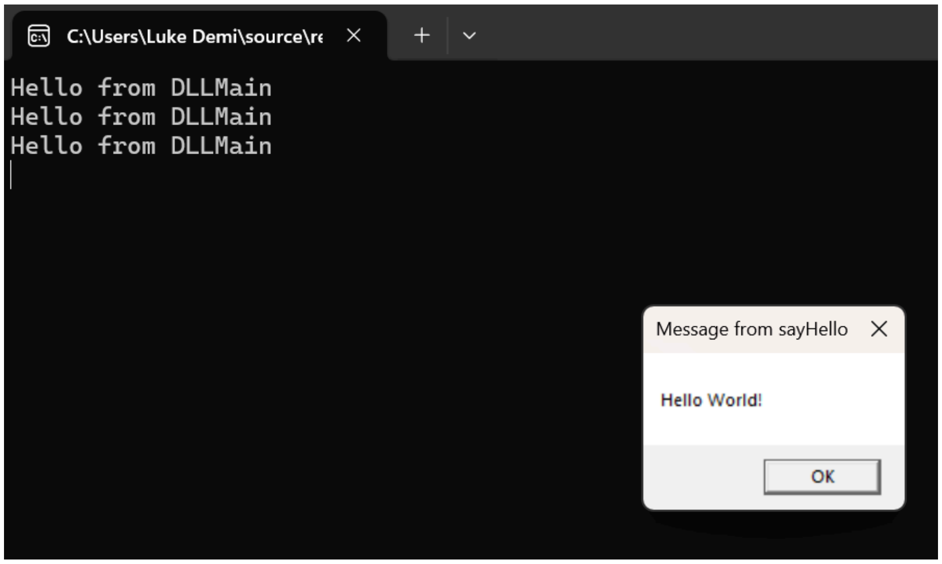
       https://learn.microsoft.com/en-us/windows/win32/dlls/dynamic-link-library-security

Self Assessment

- Everything in this program works as expected, with the DLL's being loaded when the main executable is run for both Step 1 and Step 2 of the assignment. The only thing that seems somewhat off about my program is how when the Step 1 DLL is run, it prints the "Hello from DLLMain" message around 5 times, and I'm unsure whether this is an issue with my coding or if this is just how DLL's work. Besides that, I'm fully confident that my code fully works.

Screenshots

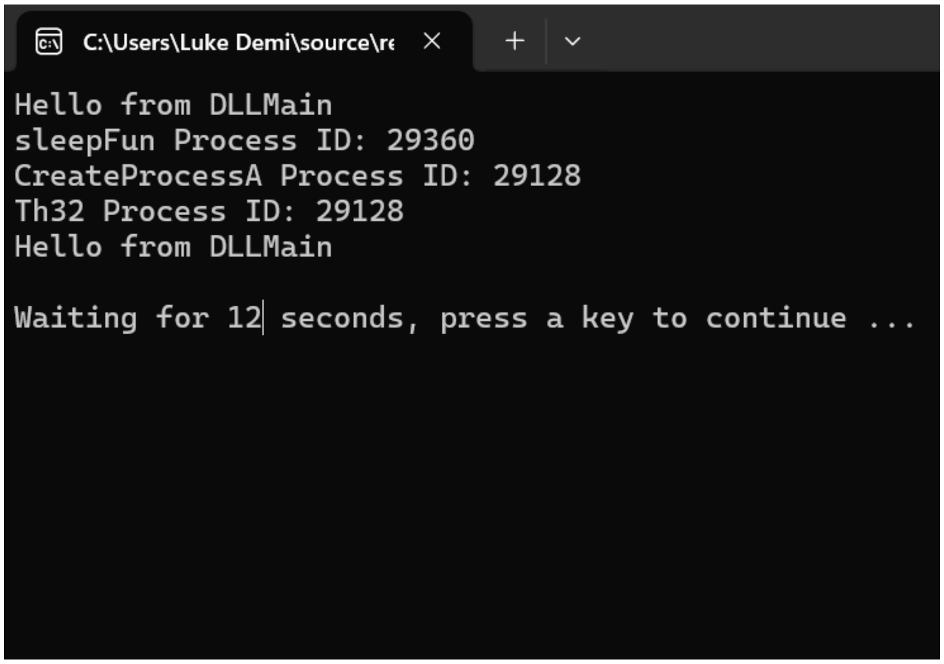
Step 1:



```
Hello from DLLMain
Hello from DLLMain
Hello from DLLMain
```

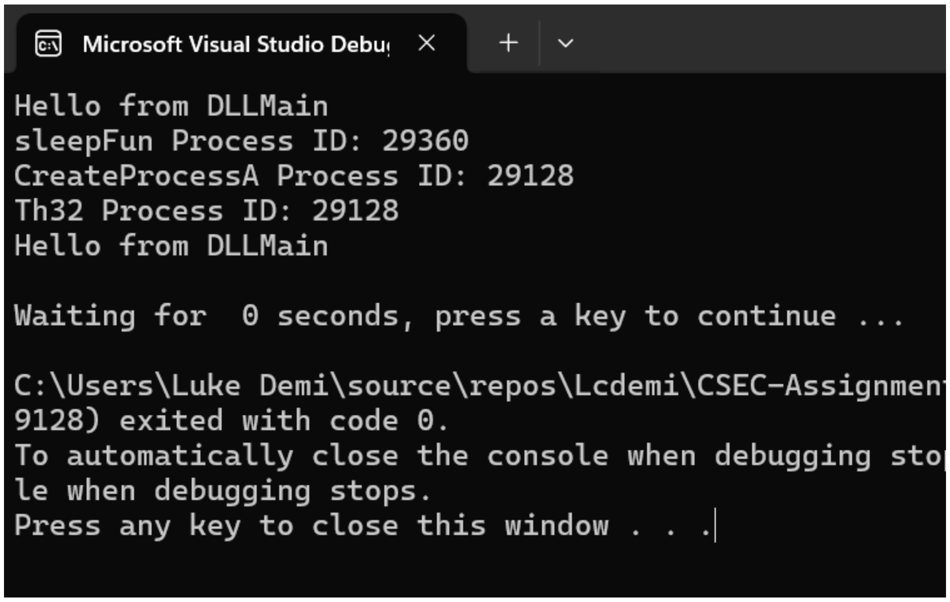
Hello World!

Step 2:



```
Hello from DLLMain
sleepFun Process ID: 29360
CreateProcessA Process ID: 29128
Th32 Process ID: 29128
Hello from DLLMain

Waiting for 12 seconds, press a key to continue ...
```



```
Hello from DLLMain
sleepFun Process ID: 29360
CreateProcessA Process ID: 29128
Th32 Process ID: 29128
Hello from DLLMain

Waiting for  0 seconds, press a key to continue ...

C:\Users\Luke Demi\source\repos\Lcdemi\CSEC-Assignmen
9128) exited with code 0.
To automatically close the console when debugging sto
le when debugging stops.
Press any key to close this window . . .
```